BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

FIG. 1

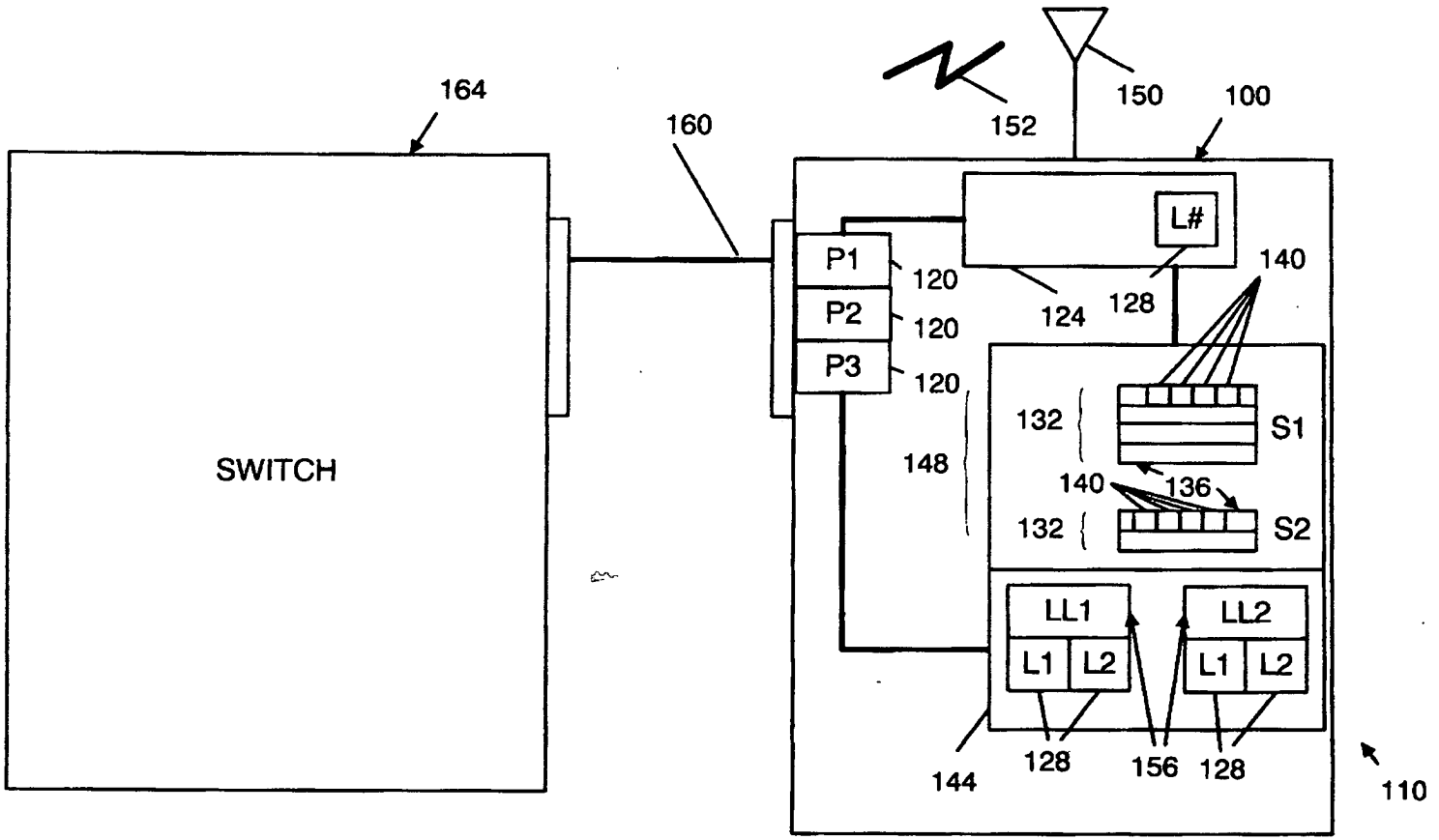
164
160
152
150
100
SWITCH
L#
P1
P2
P3
120
120
120
124
128
140
132
S1
148
140
136
132
S2
LL1
LL2
L1
L2
L1
L2
144
128
156
128
110

FIG. 2

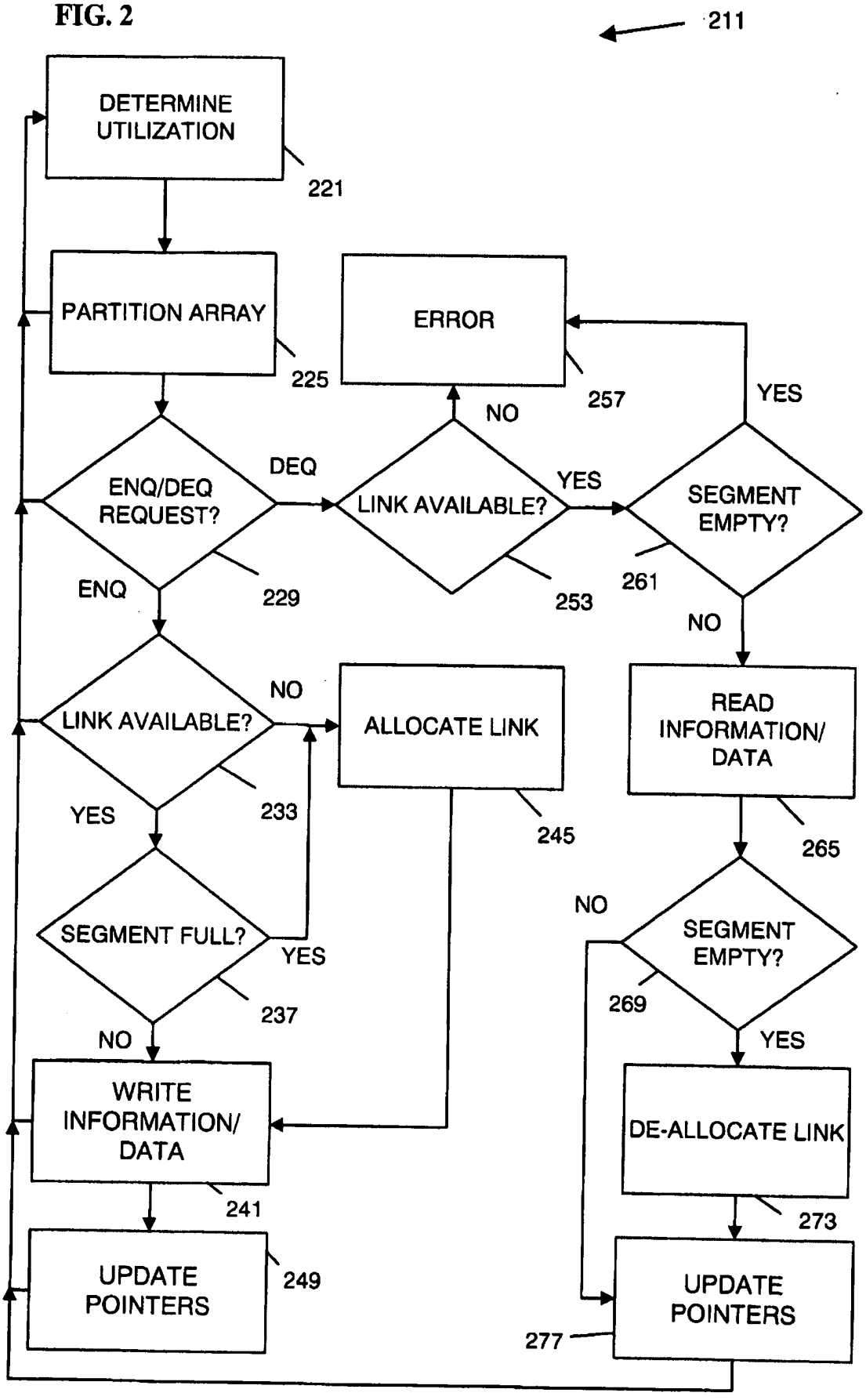
211
DETERMINE UTILIZATION
221
PARTITION ARRAY
225
ENQ/DEQ REQUEST?
229
DEQ
ENQ
LINK AVAILABLE?
253
NO
YES
ERROR
257
SEGMENT EMPTY?
261
YES
NO
LINK AVAILABLE?
233
NO
YES
ALLOCATE LINK
245
SEGMENT FULL?
237
YES
NO
WRITE INFORMATION/ DATA
241
UPDATE POINTERS
249
READ INFORMATION/ DATA
265
SEGMENT EMPTY?
269
NO
YES
DE-ALLOCATE LINK
273
UPDATE POINTERS
277

FIG. 3

```
ENQ_OPERATION
// This is a pseudo code for adding an entry into the link list.
// WP - Write pointer
// Info_array : The informantion array
// cur_port_id : the ID of the port for whcih the enqueue operation
//                    is being done. Note that many ports are sharing this
//                    resource.
// LR - Link RAM, linked list information stored here
// This pseudo code is for a 4 to 1 linked list                                  } 371

If (WP[cur_port_id][1:0]==2'b00) {                                               — 373
// If the lower 2 bits of the WP of the current port is 0, a new segment
// is required for this port.
// Take the free_avail_link that provides a pointer to the first free segment
// and make the WP point to the next location (01) in the new segment.
// store the information in the 1st location (00) of the new segment.
  WP[cur_port_id] <= {free_avail_link, 2'b01};
  Info_array [{free_avail_link, 2'b00}] <= Info;                                 — 375
} Else {
// If the lower 2 bits of the WP are non zero, it means that the segment has
// space to store some more information. Keep adding the information in the
// empty locations of the segment. Note that the information storage is
// sequential within a segment.
  WP[cur_port_id] <= {WP][cur_port_id][11:2], WP[cur_port_id][1:0]+2'b01};        — 377
  Info_array[WP[cur_port_id]] <= Info;
}

// This portion keeps track of whether the link list for a port is empty
// or not. It also updates the linked list with the new segments.
if (!Empty_Flag[cur_port_id])
   { if (WP[cur_port_id][1:0]==2'b00)
            LR[WP[cur_port_id][11:2]] <= free_avail_link;                        — 379
// if the linked list for the current port is not empty then whenever
// a new segment is allocated, link it in the link ram of the current
// port.
}
Else
// if the linked list is empty for a particular port then initialise the
// read and write pointers for this. Also reset the empty flag.
// Note that the read pointer will be updated by the dequeue operation
   { Empty_Flag[cur_port_id] <= FALSE;                                           — 381
     if (WP[cur_port_id][1:0]=2'b00)
          RP[cur_port_id] <= {avail_link_S0, 2'b00};
   }

DEQ_OPERATION
// this is the pseudo code for the dequeue operation.

Cur_rp = RP[cur_port_id];                                                        — 383
// generation of the empty condition for the dequeue operation
Empty_condition = WP[cur_port_id]=={Cur_rp [11:2], Cur_rp[1:0]+2'b01}
If (~Empty_condition && Cur_rp[1:0]==2'b11 ) {                                   — 385
// if the linked list is not empty and we are      reading the last location
// within a segment, then take the read pointer from the link ram.
  RP[cur_port_id] <= {LR[RP[cur_port_id][11:2]], 2'b00};
} else {
// if we are not reading the last location within a segment then keep on
// incrementing the read pointer within the segment.
  RP[cur_port_id] <= {RP[cur_port_id][11:2], RP[cur_port_id][1:0] + 2'b01};       — 387
}

// When the segment is completely read, put the free segment in the pool of the
// free segments.
If(RP[cur_port_id][1:0]=2'b11)                                                   — 389
      Put_free_link (RP[qnum_s0][11:2]); // LR write

// Set the empty flag whenever the empty condition is detected for a particular
// port
If (Empty_condition)   Empty_Flag[cur_port_id] = TRUE;                           — 391
```

FIG. 4